Intelligent Information Management, 2015, 7, 59-69
Published Online March 2015 in SciRes. http://www.scirp.org/journal/iim
http://dx.doi.org/10.4236/iim.2015.72006

Inferring Locations of Mobile Devices from Wi-Fi Data

Leon Wu, Ying Zhu

Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, Oshawa, Canada
Email: leon.wu@uoit.ca, ying.zhu@uoit.ca

Received 23 January 2015; accepted 12 February 2015; published 5 March 2015

Abstract

Mobile phones are becoming a primary platform for information access. A major aspect of ubiquitous computing is context-aware applications which collect information about the environment that the user is in and use this information to provide better service and improve user experience. Location awareness makes certain applications possible, e.g., recommending nearby businesses and tracking estimated routes. An Android application is able to collect useful Wi-Fi information without registering a location listener with a network-based provider. We passively collected the data of the IDs of Wi-Fi access points and the received signal strengths. We developed and implemented an algorithm to analyse the data; and designed heuristics to infer the location of the device over time—all without ever connecting to the network thus maximally preserving the privacy of the user.

Keywords

Location Awareness, Mobile Device, Data Analysis, Data Inference

1. Introduction

The proliferation of Wi-Fi networks has come to the point that under normal everyday circumstances, a mobile device being carried around by an individual is able to detect a set of Wi-Fi access points at almost any given time in the day. The user may not be able to connect to the majority of these Wi-Fi networks, nor is it likely that she would wish to, as that would make her location visible and known—privacy would not be preserved. However, information of these ubiquitous wireless access points can still be obtained, such as BSSID, SSID and received signal strengths, which we propose to collect, store in a structured way, and analyze algorithmically to infer the locations of the device over time.

We consider the following scenario. A commuter takes the same train daily and during the extended period of

time on the train, she may have the habit of taking a quick nap—this may cause her to miss her station. It would be useful if her mobile device could keep track of the sets of wireless networks detected over time (the same pattern everyday) and infer her locations, and trigger an alert when the train is near her destination.

In our work, we collect the Wi-Fi access point data and store them in a structured way in the on-board database; and then analyze this data using some heuristics to make location inference. The heuristics we employ include defining a "distance" metric between two sets of data from detected wireless networks at different time points, in order to obtain a measure of similarity between them. The similarity measure allows us to infer relative locations.

Location awareness using Wi-Fi information on mobile applications can provide significant assistance for travelers to manage their time on the road and improve travel efficiency. Most importantly it can reduce the cost of purchasing a data plan to obtain the real time information. For example, there is no need to use data service from major service provider to keep the mobile device connected with Internet. It can also be used to track their devices if the location based database information is stored on a dedicated server or in the cloud. Further enhancements can include location inference towards calculating distances while on the road.

Although GPS (Global Positioning System) already provides the location information for the mobile device, keeping GPS running on the device consumes substantial battery power, in contrast to the passive detection of Wi-Fi access points. Using Wi-Fi information to determine locations on mobile devices significantly reduces battery consumption compared to using GPS.

2. Related Work

An active part of the research in ubiquitous computing has been in location-aware computing [1]. There has been a great deal of interest in context-aware applications [2] and the location of devices on which the applications are executing is one of the main components of context for an application. There have been many works, e.g., [3]-[13], in the inference of in-door locations of mobile devices using the signal measures surveyed by the device in the environment of a wireless LAN. These works share the common broad approach of analyzing measured signal strengths. The methods and techniques used include: error minimization (lacks robustness in presence of noisy data), multilateration (problematic with obstructive objects), probabilistic Bayesian models, triangulation, probabilistic fingerprinting method. All these related works are interesting and offer accuracy in localization. However, they differ from our work in that they are only concerned with localization of a device in an indoor space, usually just one room. Some of them even require the prior knowledge of exact coordinates of access points which is a stringent requirement that we do not impose for our service to work.

Recent work on radio map-based techniques may be categorized into deterministic ones, e.g., [4] [14] and probability distribution-based ones, e.g., [3] [9] [15] [16]. In both [4] and [14], the a priori knowledge of the physical positions of all access points in a given area are used to generate a function that maps signal strengths to physical distance. This function is obtained empirically from a set of trained points and observations. The function is then applied to real-time observed signal strengths to generate predictions of locations. The authors of [15] take a machine learning approach to the problem of location estimation. A Bayesian-based probabilistic model is developed to describe the distribution of received signal strengths at various locations, which can then be used to infer the location from real-time observed signal strengths.

3. Methodology

The open source nature of the Android development environment provides incentives for amateur and professional developers to make contributions to their Java-based application development and design their own versions of Android for their own needs. The size of the Android developer community has rapidly increased since the first Android powered phone was released in 2008. Android applications are written primarily in the Java language, and have attracted a large number of Java developers to join forces to extend the functionalities of the devices.

There are two phases in this research; they are shown in **Figure 1**. The first phase is the development of the mobile application that collects Wi-Fi information and stores that data in the database on the mobile devices. The second phase is the data analysis that leads to location inference. It includes database creation, data inference heuristics, data plotting, data manipulation and data query. Both parts are closely related to complete the location awareness using Wi-Fi information.

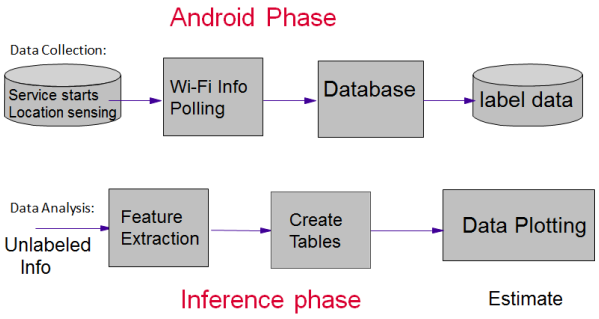


Figure 1. Overall development phases.

3.1. Android Development

In this Android development research, the application has involved most of the above components with different approaches. A flow diagram is presented in **Figure 2** to demonstrate how the Android development flows from the start till the end. The application has one activity to integrate with service if it is not already running. The user can communicate with the service through a simple user interface that service exposes. The android application is developed using API 10 Gingerbread Android 2.33. Citing from some unofficial reports a few months ago, the majority of android mobile phone users are still using the older Android OS in the current market. Gingerbread has arrived during the peak time of Android development.

3.2. Data Inference

Data inference is the second stage of this research project. After collecting all the Wi-Fi information including ssid, bssid and rssi, the data can be analysed offline. The purpose of the data inference is to plot the history path of the corresponding nodes and connect the Wi-Fi logical space to physical space which is the known addresses.

There are two ways of polling the database information from the mobile device. One is to register as the super users to the mobile device, and then the super user would easily reuse the database information or translate it into text format. The second approach is the Java implementation at application level to store the database information as text format on the SD card of the mobile device. This research is opted for both options as the mobile application development has gone to some levels in depth.

In order to select meaningful data and then interpret them as different nodes, the techniques of grouping and calculation of similarity between nodes are used. The diagram in **Figure 3** can be referred to as the flow chart for data manipulation from the data collection. There are steps that could be key points to determine the readings from the data file.

The programming language used for data inference is Python using DB-API 2.0 interface with SQLite Database. There are 5 tables created in the database from the raw data for the data inference. The future step can be developed to the stage of regrouping the database at runtime. Raw data collection file has one large table in the database. It contains five columns id, time, ssid, bssid and rssi. In order to efficiently use “raw” information from the database, the data inference is used at this stage to manage the data collection.

4. Android Development

Google provides graphical development environments based on the Eclipse IDE to develop applications. Android Developer Tool (ADT) is based on the Eclipse IDE and provides additional functionality to develop Android applications. Android platform and developer tools are excellent for programming mobile devices. Android applications are primarily written in the Java language. The compiled Java code is compressed with an archive file with a suffix of apk. This is the file that can be distributed in many ways for installing applications on mobile devices.

4.1. Android Application Architecture

In this structure, we have developed 5 classes during the Android programming development.

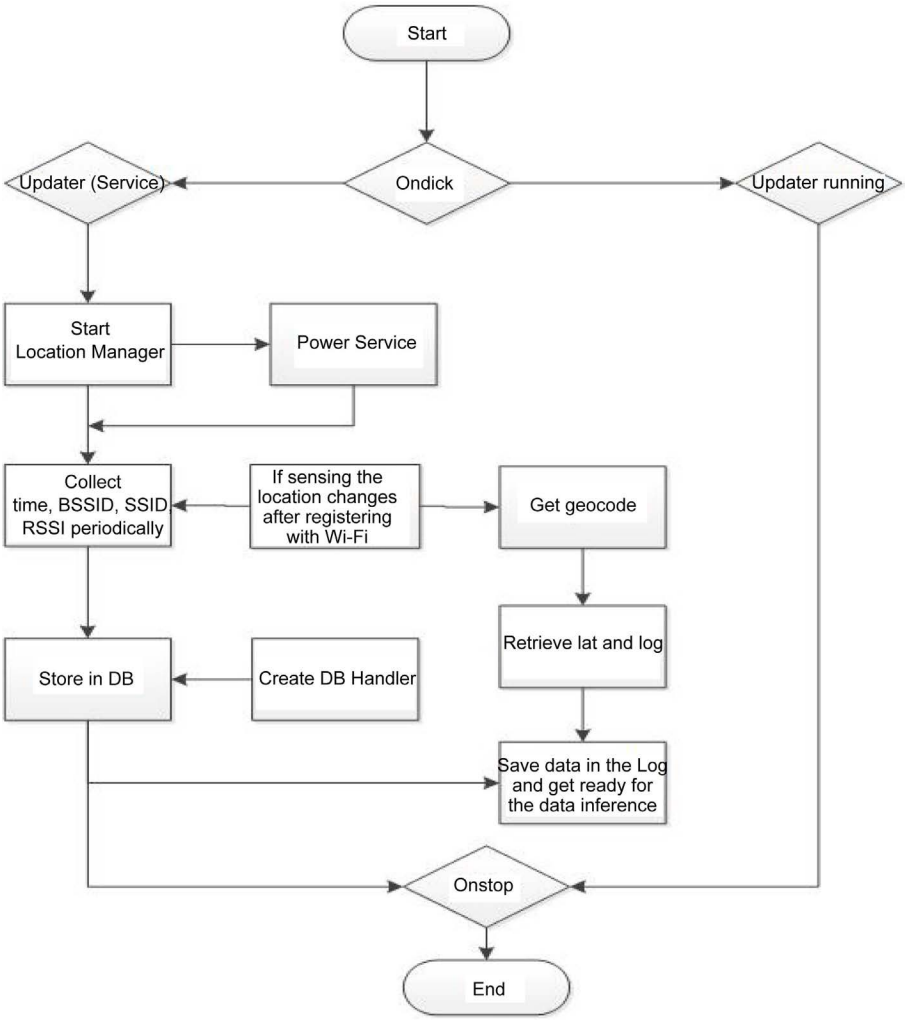


Figure 2. Flow chart of Android development.

- Access Point handles all initializing objects with constructors of the Wi-Fi information such as Service Set ID (ssid), Business Service Set ID (bssid) and signal strength. ssid is a string that indicated as access point's name. bssid is a 6 byte vendor assigned unique MAC address. It also sets the methods to set and retrieve all these Wi-Fi information.
- Database Handlder takes care of database creation and upgrade. It executes basic SQL statements such as insertion, retrieval and updating.
- Leon Wifi ServiceL is the main activity piece that created simple button for users to start the service.
- mLocation Receiver contains the add-ons that sensing the location changes and store them into the database.
- mService is the main force to start the service running in the background and collect Wi-Fi information periodically and store them into the database file on the SD card.

4.2. Detailed Approaches

- Service based application. There are a number of reasons why service based programming is suitable for this research. First, the application should handle operations such as collecting the Wi-Fi information silently without a User Interface. Android accords services a higher priority than inactive or invisible activities [17]. Second, unlike activities which present a rich graphical interface, services run in the background, thus prolonging battery life. Third, the service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other

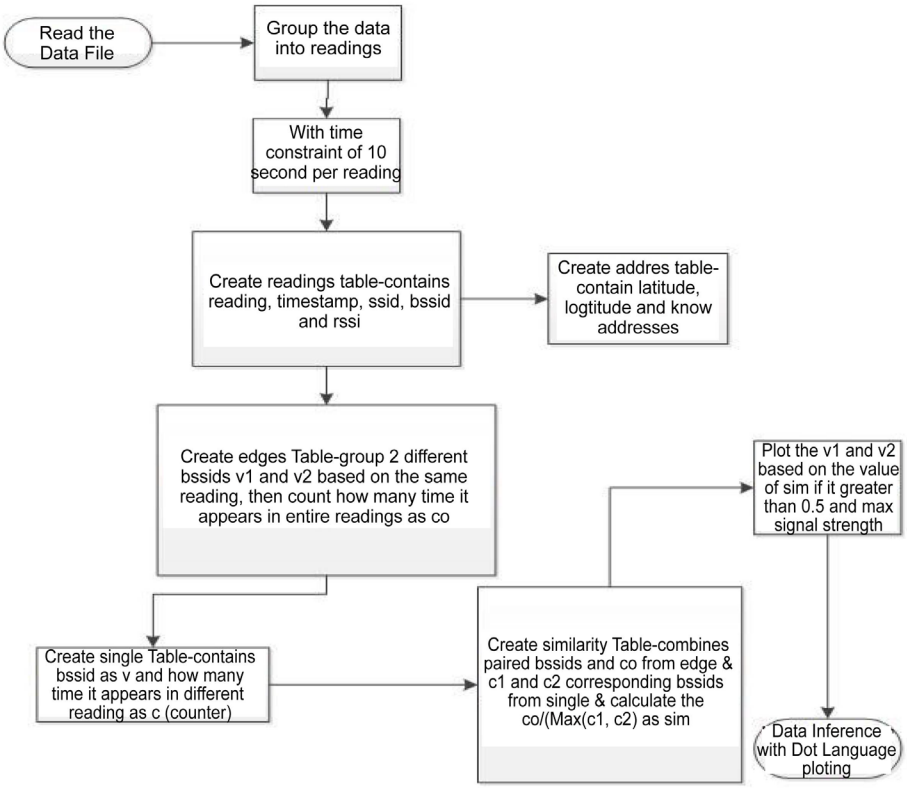


Figure 3. Flow chart for data analysis.

applications to use [3].

- Wakelock. Because Android smart phone Galaxy S has the screen timeout that prevent the service base tracking application collecting data while the screen is blackout, therefore the power management mechanism must kick in to allow the service based application collecting Wi-Fi information. A wake lock mechanism is used to make sure the device stays on.
- Simple Thread. A new simple thread has been chosen to pull data from Wi-Fi environment in order to focus on the need to interact with network. Static method of Thread.sleep() provided the solution to one of main objectives in this application—periodically polling the Wi-Fi data.
- SQLite Database. Android system equips with SQLite which has been available on the platform so the application could manage its own private database.
- Location Sensing. The method of using pending intent object with a Broadcast Receiver class is added to keep track on the changing location information such as latitude and longitude obtained from registered Wi-Fi connection.
- Location Report. In order to verify the physical location from collected data, the location Geocode is introduced to interpret latitude and longitude information into address information.

5. Data Inference

Data inference is second stage of this work. After collecting all the Wi-Fi information including ssid, bssid and rssi, the data can be analyzed offline. The purpose of the data inference is to plot the history path of the corresponding nodes and connect the Wi-Fi logical space to physical space which is the known addresses.

5.1. Database Manipulations

A new database is generated for easy access data inference with the manipulation and calculation based on time stamps. The algorithm of these tables could be explained in a nutshell. The readings table is created first from the raw data that contains id, timestamp, ssid, bssid and rssi. The algorithm of creating the readings table is to

group the data into different blocks based every 10 seconds using the timestamp. Next, the edge table and single table are generated based on readings table to select 2 unique access points and count them both and individually how many times they appear in the entire readings table. Similarity table is the key table for calculating how similar these two access points in the entire readings table. Android application is capable of obtaining physical address after registering with Wi-Fi connections; therefore the useful physical address table is also created from the raw data. In order to connect the Wi-Fi logic distances to physical known addresses, the heuristics of distance and multi-hop are explained in greater details in the subsections.

Readings Table. Instead of hundreds, thousands of the sequential data information appending on each time stamps with one second apart, the readings block table could be used systematically as the first step to group the data every 10 seconds per reading in the readings table in the new database file. An example of the readings table is shown in **Figure 4**, after regrouping data according to the time constraint.

Address Table. The Address table contains information of bssid and corresponding longitude, latitude and physical address, converted from the Android application when the mobile device is registered with the Wi-Fi connection. It could be useful for data inference especially for data retrieval between Wi-Fi space and physical space and then the data plotting.

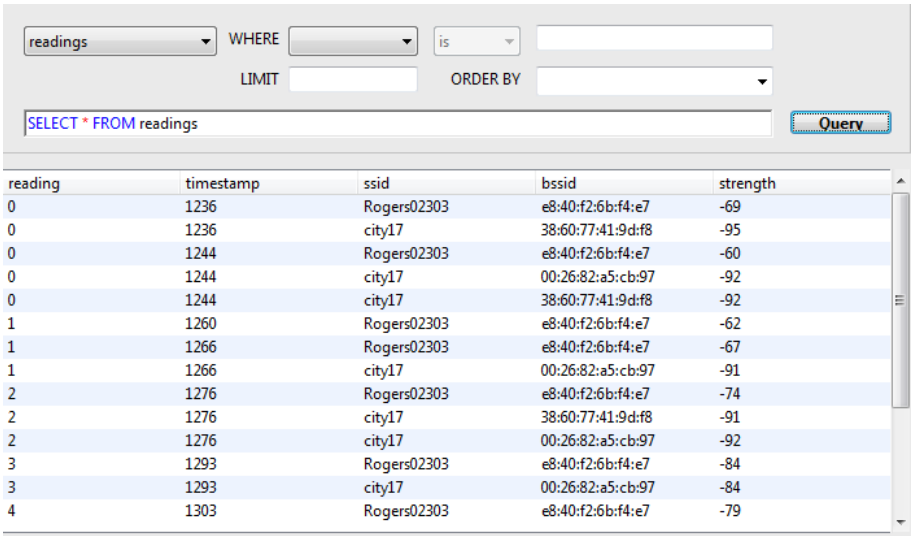
Edge Table. In order to group a pair of different bssids as nodes for data inference especially for data plotting, the edge table was created for this purpose. The edge table contains a pair of different bssids referred as v1 and v2 co-occurring in the same reading and then count how many times they co-exist in entire readings as co. Therefore, the same reading of 2 different bssids is paired in one row joining with the number of times they appear from different readings.

Single Table. The reason why table single was named is to eliminate the redundancy of a bssid appear multiple times in the entire readings. Single table contains each bssid as v in readings and counter as c that counts how many time v appears in different reading in entire readings table. It collects statistical information of each individual bssid.

Similarity Table. The similarity table combines edge and single tables and a few other additional important heuristics calculations and definition columns. The first three columns are the pair of bssids as v1 and v2 and co-occurrence counter from edge table. The fourth and fifth columns are the counters of c1 and c2 counters that count how many corresponding v1 and v2 appear in reading in the entire readings table. The sixth column is the similarity heuristics calculation as sim about the paired v1 and v2. The seventh column is the Wi-Fi space distance heuristic calculation as dist between v1 and v2.

5.2. History Path Plotting

Data plotting is the connection of trace of combination of printing v1 and v2 from the table of similarity and



reading	timestamp	ssid	bssid	strength
0	1236	Rogers02303	e8:40:f2:6b:f4:e7	-69
0	1236	city17	38:60:77:41:9d:f8	-95
0	1244	Rogers02303	e8:40:f2:6b:f4:e7	-60
0	1244	city17	00:26:82:a5:cb:97	-92
0	1244	city17	38:60:77:41:9d:f8	-92
1	1260	Rogers02303	e8:40:f2:6b:f4:e7	-62
1	1266	Rogers02303	e8:40:f2:6b:f4:e7	-67
1	1266	city17	00:26:82:a5:cb:97	-91
2	1276	Rogers02303	e8:40:f2:6b:f4:e7	-74
2	1276	city17	38:60:77:41:9d:f8	-91
2	1276	city17	00:26:82:a5:cb:97	-92
3	1293	Rogers02303	e8:40:f2:6b:f4:e7	-84
3	1293	city17	00:26:82:a5:cb:97	-84
4	1303	Rogers02303	e8:40:f2:6b:f4:e7	-79

Figure 4. Readings table.

taking the maximum signal strength from the table of readings. In order to clean up the unwanted nodes for the plotting, the path is taken from the selective pair of v1 and v2 after comparing their similarity values are great than 0.5. It is in line with the heuristic Wi-Fi distance calculation as the user defined value. If value of sim is chosen only greater than 0.0, the majority of the pair of v1 and v2 will be added into the path.

The following plot is that derived one particular data file called drive.txt that has been collected early. Blue line is the link between nodes. Black dot represents the selected node. Those black lines and dots are noise from the surrounding environments such as passing by mobile users, and they are not part of the history path.

Figure 5 below shows the plot of the path of a trip taken from the UOIT Campus to Regent Centre, Downtown Oshawa and then come back to UOIT campus in various buildings with selective labels. In **Figures 5-10**, we shown experiments of collecting Wi-Fi information on various trips, then analyzing the data to plot the routes

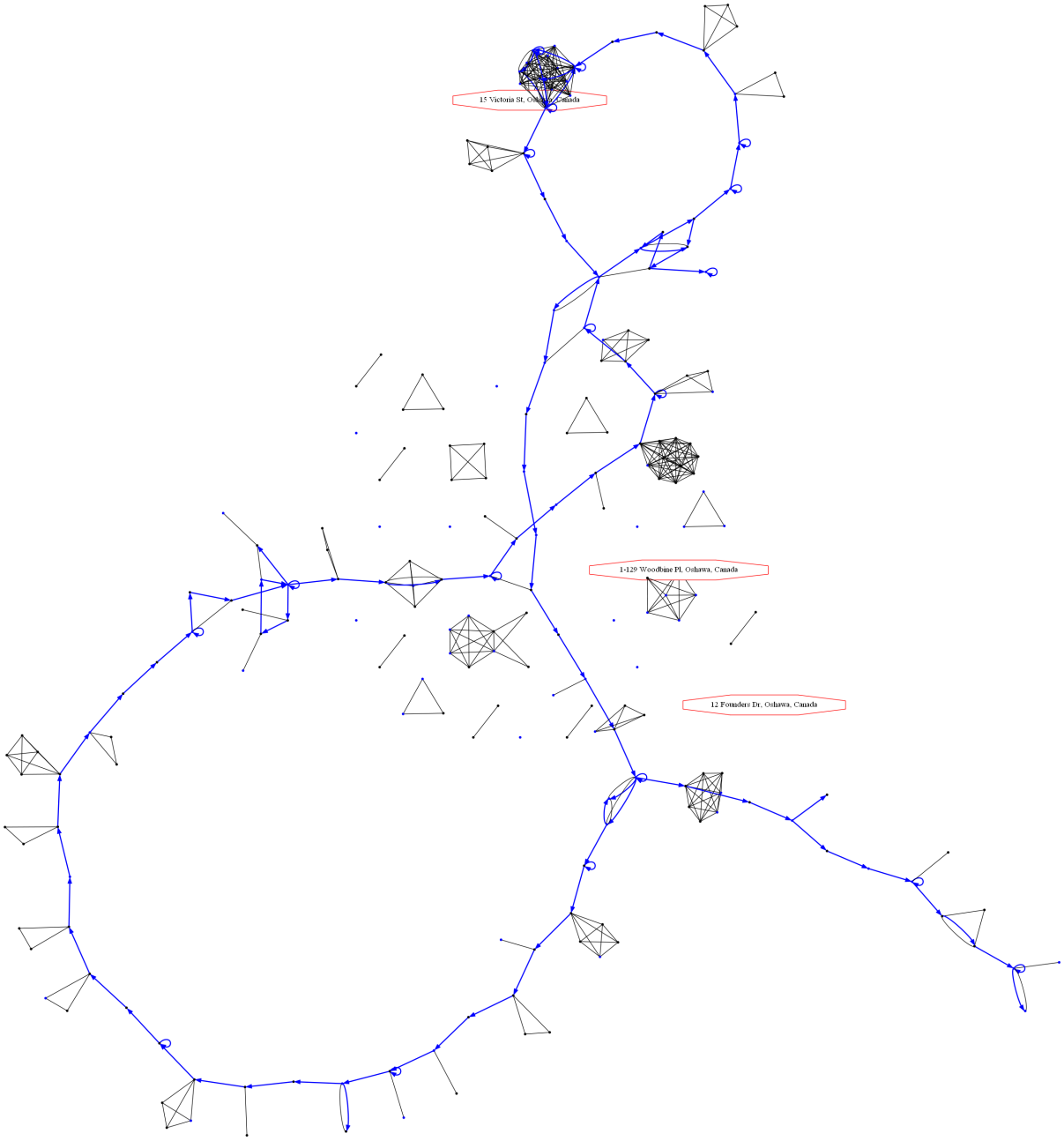


Figure 5. Data collection route 1.

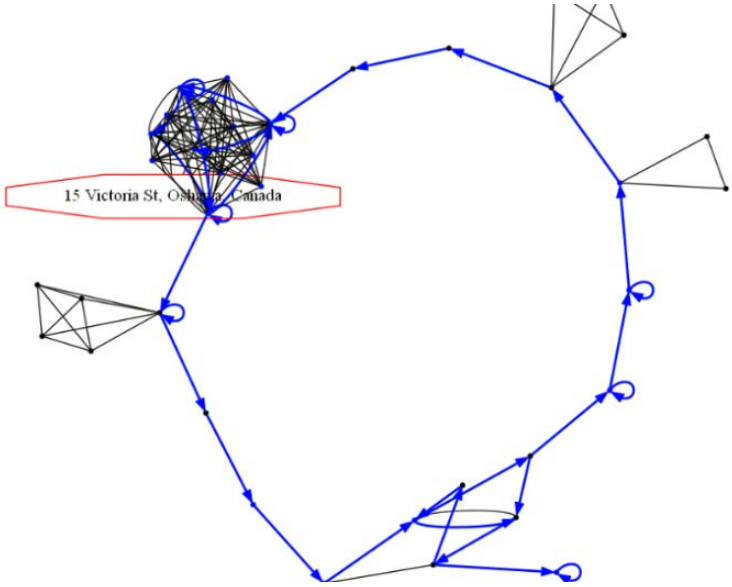


Figure 6. Route 2 top zoomed in with labelled address.

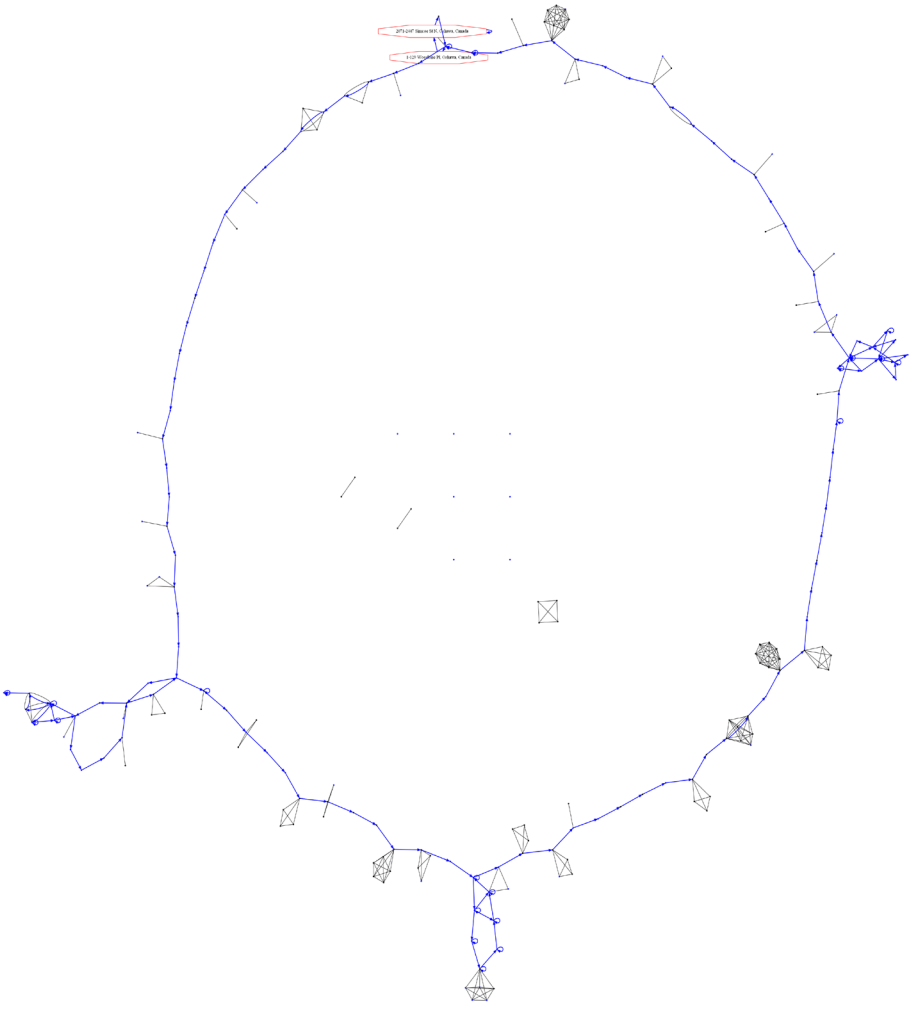


Figure 7. Data collection route 2.

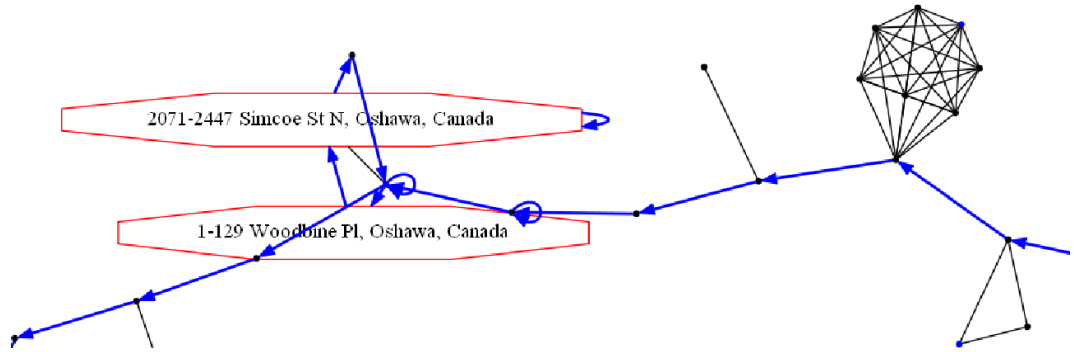


Figure 8. Route 2 with labels of known addresses zoomed in.

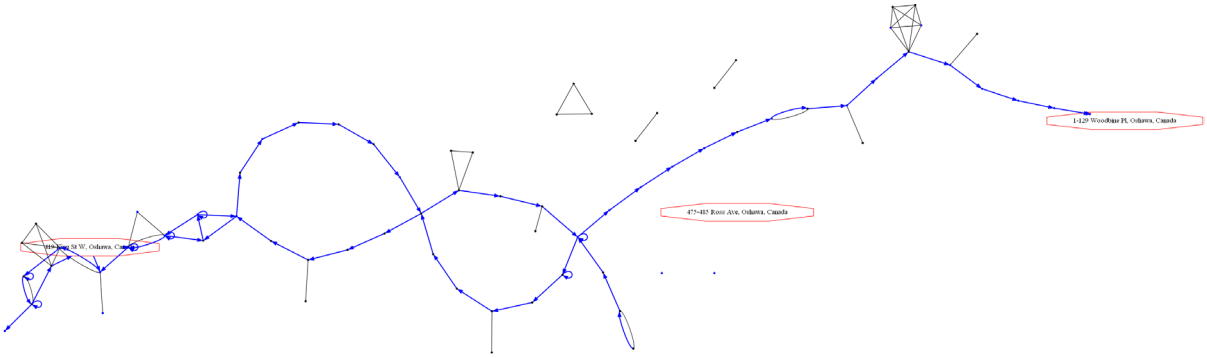


Figure 9. Data collection route 3.

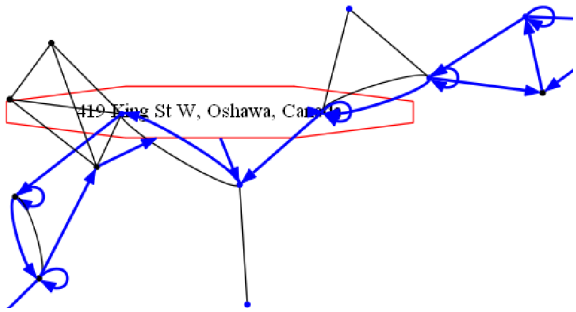


Figure 10. Route 3 with labels of known addresses zoomed in.

taken by the user. Some show the zoomed-in parts of the route where we also label the nodes with known postal addresses.

5.3. Similarity and Wi-Fi Space Distance Metrics

This portion is mainly focus on the calculation of computing a measure between 0 and 1 to determine how similar v1 and v2 are with respect to the readings. If they appear together all the time, then they are very similar (sim = 1), but if they never appear together, then they are not similar at all, so sim = 0. The threshold is set to 0.5 to increase the clarity of the plotting. The formula to achieves this goal use as (co)/max(c1,c2). The node distance calculation is done based on similarity calculation. The distance d1 can be presented as,

$$d1(x) = (1 - \alpha 1X) * K1 \text{ where K1 is user define value as 100 refers as far away} \quad (1)$$

$$d2(x) = (1 - \alpha 2X) * K2 \quad (2)$$

where d1 and d2 are distance between two nodes v1 and v2 and X is the similarity between the two nodes v1 and v2.

Substitute K1= 100 and X = 0.5 into Equation (1), d1(x) = 100(1 − α1) = 1, so α1 can be easily obtained 99/50, therefore

$$d1(x) = 100 * (1 - 1.98\ X) \tag{3}$$

Similar technique can applied to obtain α2 using the graph shown above

$$1 = (1 - \alpha 2X) * K2 \quad \text{and} \quad 0 = (1 - \alpha 2) * K2$$

Therefore K2 and α2 can be obtained as 2 and 1 respectively.

$$d2(x) = 2 * (1 - X) \tag{4}$$

The definition on similarity and calculation of "distance" between two nodes can be obtained from Equations (1), (2), (3), and (4). The K1 value could be chosen differently, but the plotting results remain the same because the history path is not based on the distance between nodes. The logical distance is built on to show users how far from the position to a known address. With more extensive data collection, the real distance values could be introduced in the future. We show the plot of the distance metric versus the similarity measure in **Figure 11**.

5.4. Data Query

Data query returns logic distance values from the known addresses in an ascending order. It has not been developed at this stage to explore how accurate the distance values are in terms of real distance, therefore there is no unit for the distance values. By injecting difference threshold of similarity, distance values might return differently from the data query, but it is insignificant at this stage because the logical distance values have not been measured with real distance values. However, it has demonstrated the huge potential to link the logical distance values with the real distance values using Google maps if further investigation and development are put on the agenda.

5.5. Discussion of Usefulness of Location Inference from Wi-Fi Information

We have conducted several experiments to evaluate the usefulness of our implementation of location inference using Wi-Fi information. In **Figures 5-10**, we show the three routes that we have taken (driving in a car) to collect Wi-Fi data. Based on the raw Wi-Fi data, *i.e.*, the sequence IDs of Wi-Fi access points, we use our implementation to analyze the data by calculating the similarity measure and distance metric. The distance metric provides the logical distances inferred between the location spots traversed along these routes. Even though the distances are logical and not physical, plotting the route still gives useful information of the paths that the user has taken and the places that the user has visited over time, especially if some of the locations visited could be labeled with actual postal addresses, whether using historical data or user input or occasional GPS querying.

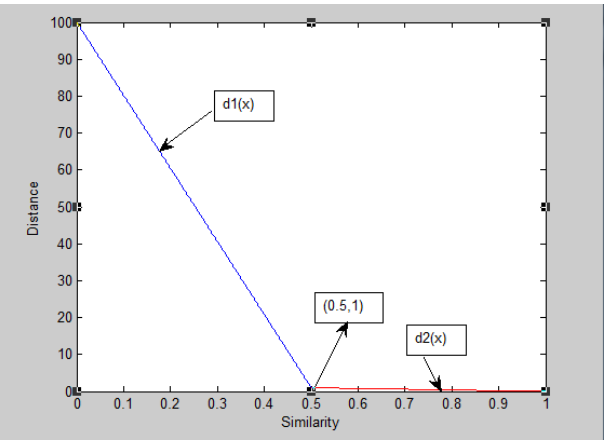


Figure 11. Distance heuristic.

6. Conclusion

In summary, location awareness is one of most popular researches among all mobile application development. It brings the convenience to mobile users especially on the go. The android application collecting Wi-Fi information provides the one of most economical ways to present the location awareness mechanism. As we know, mobile devices equipped GPS sensors are the power source driven for location display to perform efficiently and accurately. This research approach can significantly resolve the issue of utilizing power resource on the mobile devices by sensing only Wi-Fi information.

Acknowledgements

The authors are grateful to the Natural Sciences and Engineering Research Council of Canada (NSERC) for the funding that supported this work.

References

[1] Lock (2013) Documentation on the java.util.concurrent.locks.Lockclass. http://developer.android.com/reference/java/util/concurrent/locks/Lock.html

[2] Schilit, B.N., Adams, N. and Want, R. (1994) Context-Aware Computing Applications. 1*st Workshop on Mobile Computing Systems and Applications*, *IEEE*, *WMCSA* 1994, 85-90.

[3] Castro, P., *et al.* (2001) A Probabilistic Room Location Service for Wireless Networked Environments. Ubicomp 2001: Ubiquitous Computing. Springer Berlin Heidelberg, Berlin.

[4] Bahl, P. and Padmanabhan, V.N. (2000) RADAR: An In-Building RF-Based User Location and Tracking System. *INFOCOM* 2000, 19*th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings of IEEE*, **2**, 775-784.

[5] Yang, J. and Chen, Y.Y. (2009) Indoor Localization Using Improved rss-Based Lateration Methods. *Global Telecommunications Conference*, *GLOBECOM* 2009, Honolulu, 1-6.

[6] Gwon, Y., Jain, R. and Kawahara, T. (2004) Robust Indoor Location Estimation of Stationary and Mobile Users. *INFOCOM* 2004, 23*rd Annual Joint Conference of the IEEE Computer and Communications Societies*, Hongkong, 1032-1043.

[7] Small, J., *et al.* (2000) Determining a User Location for Context Aware Computing through the Use of a Wireless LAN Infrastructure. Project Aura Report. http://www.cs.cmu.edu/~aura/docdir/small00.pdf

[8] Grossmann, U., Schauch, M. and Hakobyan, S. (2007) RSSI Based WLAN Indoor Positioning with Personal Digital Assistants. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. 4*th IEEE Workshop on IDAACS*, Dortmund, 653-656.

[9] Ladd, A.M., *et al.* (2005) Robotics-Based Location Sensing Using Wireless Ethernet. *Wireless Networks*, **11**, 189-204.

[10] Retscher, G., *et al.* (2006) Performance and Accuracy Test of the WLAN Indoor Positioning System "ipos". *Proceedings of the* 3*rd Workshop on Positioning, Navigation and Communication*, University of Hanover, Germany.

[11] Teuber, A. and Eissfeller, B. (2006) WLAN Indoor Positioning Based on Euclidean Distances and Fuzzy Logic. *Proceedings of the* 3*rd Workshop on Positioning, Navigation and Communication*, University of Hanover, Germany, 159-168.

[12] Quan, M., Navarro, E. and Peuker, B. (2010) Wi-Fi Localization Using RSSI Fingerprinting. California Polytechnic University, Technical Report.

[13] Prasithsangaree, P., Krishnamurthy, P. and Chrysanthis, P.K. (2002) On Indoor Position Location with Wireless LANs. *Proceedings of the* 13*th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Lisbon, 15-18 September 2002, 720-724.

[14] Smailagic, A. and Kogan, D. (2002) Location Sensing and Privacy in a Context-Aware Computing Environment. *IEEE Wireless Communications*, **9**, 10-17. http://dx.doi.org/10.1109/MWC.2002.1043849

[15] Roos, T., Myllymäki, P., Tirri, H., Misikangas, P. and Sievänen, J. (2002) A Probabilistic Approach to WLAN User Location Estimation. *International Journal of Wireless Information Networks*, **9**, 155-164. http://dx.doi.org/10.1023/A:1016003126882

[16] Youssef, M.A., Agrawala, A. and Udaya Shankar, A. (2003) WLAN Location Determination via Clustering and Probability Distributions. *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, Fort Worth, 26-26 March 2003, 143-150.

[17] Negnevitsky, M. (2008) Artificial Intelligence: A Guide to Intelligent Systems. 2nd Edition, Pearson Education, Harlow.